GCCS/DII COE System Integration Support

DII COE ORACLE Client Utilities Segment System Administrator's Manual

February 07, 1997

Prepared for:

DISA/JEJA
ATTN: Ms. Claire Burchell
45335 Vintage Park Plaza
Sterling, VA 20166-6701

Contract Number: DCA100-94-D-0014
Delivery Order Number: 330
CDRL Number: A005

Prepared by:

Computer Sciences Corporation
Defense Enterprise Integration Services
Four Skyline Place
5113 Leesburg Pike, Suite 700
Falls Church, VA 22041

THIS DOCUMENT IS UNCLASSIFIED

Defense Information Infrastructure (DII)

Common Operating Environment (COE)

System Administrator's Manual
ORACLE Client Utilities Segment
Version 7.3.2 (Sun Sparc/Solaris 2.5)

Version 3.0

February 07, 1997

Prepared for:

DISA/JEJA
ATTN: Ms. Claire Burchell
45335 Vintage Park Plaza
Sterling, VA 20166-6701

Prepared by:

Computer Sciences Corporation
Defense Enterprise Integration Services
Four Skyline Place
5113 Leesburg Pike, Suite 700
Falls Church, VA 22041

THIS DOCUMENT IS UNCLASSIFIED

Table of Contents

Preface

The following conventions are used in this document:

Bold	Used for information that is typed, pressed, or selected in executables and instruction. For example, select connect to host.
Italics	Used for file names, directories, scripts, commands, user IDs, document names, and Bibliography references; and any unusual computer language the first time it is used in text.
Underline	Used for emphasis.
Arrows <>	Used to identify keys on the keyboard. For example <Return>.
"Quotation Marks"	Used to identify informal, computer-generated queries and reports, or coined names; and to clarify a term when it appears for the first time.
`Courier Font`	Used to denote anything as it appears on the screen or command lines. For example `User:/ora01/dba/oradb:/bin/csh.`
Capitalization	Used to identify keys, screen icons, screen buttons, field, and menu names.

1. ORACLE RDBMS Overview

The DII COE Database is based on the ORACLE Relational Database Management System (RDBMS). This section provides a general overview of the ORACLE RDBMS. For specific information and operational instructions for ORACLE user access, backup/recovery procedures, ORACLE Database Administrator (DBA) utilities, SQL*Forms, SQL*Plus, SQL*Reports, and SQL*ReportWriter access, refer to the current version of the *ORACLE RDBMS Database Administrator's Guide* and the documentation for the particular products. The *Database Administrator’s Guide* describes the DIIDB database and DIIDB database architecture, and presents detailed instructions for database administration activities. For DIIDB, the ORACLE RDBMS is installed on the database server, a SUN Solaris 2.5.1 platform sized according to the DISA Contract.

Logging onto a database server as *oradba* automatically starts the ORACLE Server Manager, a tool to browse and alter the database. In addition, an X window is available; it is iconized in the lower left corner of the screen.

To use the Server Manager, enter your Oracle DBA account name and password for the user name and select **connect**. ORACLE system information is selectable by single-clicking on the desired option, e.g., STORAGE, SECURITY, INSTANCE, RECOVERY and SCHEMA. Much of what is done using the SQL*DBA commands is discussed below. ORACLE commands can be run by selecting **New Worksheet** from the **File** menu.

2. Understanding the ORACLE Database

This section provides an overview of the ORACLE Relational Database. This information is provided as an introduction for readers unfamiliar with database concepts. The current revision of the ORACLE vendor documentation is the comprehensive reference for using and administering the DIIDB database.

An ORACLE database is a collection of data that is treated as a unit. The principal purpose of a database is to store or retrieve related information. An ORACLE database can be configured to lock out various features and portions of the database to ensure system security. The ORACLE database is accessed via an ORACLE instance (the software that manipulates the database) when the database is opened. When the database is closed, its data is unavailable to users.

The ORACLE RDBMS enables DIIDB users to maintain, monitor, and manipulate large quantities of data in a structured environment.

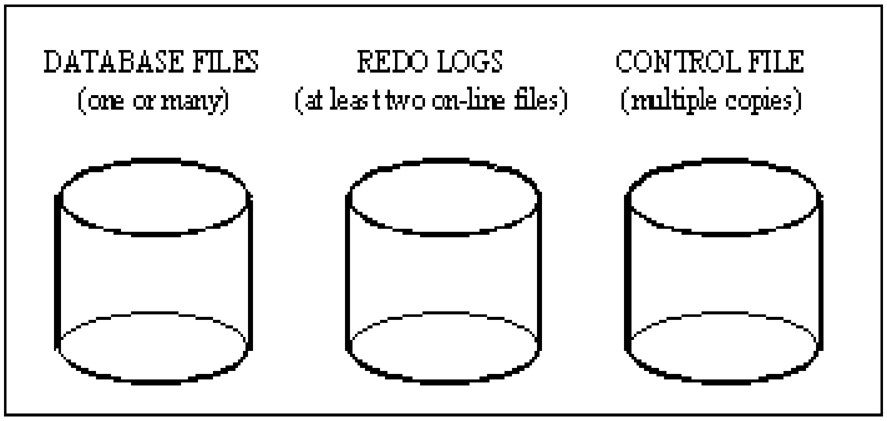
The ORACLE RDBMS offers DIIDB users many benefits:

- Easy access to all data
- High transaction-processing performance
- High, controlled availability
- Flexibility in data modeling
- Manageable security, user hierarchy, and database-enforced integrity
- Reduced data storage and redundancy

- Independence of physical and logical data design
- A high-level data manipulation language (SQL)
- Portability, compatibility, and connectivity.

3. Database Structure

An ORACLE database has both a physical structure and a logical structure. The physical storage structure shown below can be managed without affecting access to the logical storage structure.



Physical Database Structure.

3.1 Physical Database Structure

An ORACLE database's physical structure is determined by the operating system files that constitute the database. Each ORACLE database consists of one or more data files, two or more redo log files, and several copies of the control file.

DATA FILES contain all the database data. They can be added or removed from a database, as required, to meet the user's needs. A data file can only be associated with one database, and its size cannot be changed once it is created.

REDO LOG FILES are a set of files that sequentially record all changes--whether committed or uncommitted--and are used to recover data not written to the database for some reason, such as system or media failure. The process of applying the redo log is called "rolling forward" or, more generally, "recovery."

CONTROL FILES record the physical structure of the database. A control file contains the database name, names, and locations of its data files, and the timestamp of database creation. Once an ORACLE instance is started, its control file is used to identify the database and redo log files that must be opened and accessed for database operation. Control files are automatically updated by ORACLE and cannot be manually modified. One or more copies of the control file must be maintained for database backup.

3.2 Logical Database Structure

An ORACLE database's logical structure is determined by one or more storage units, called tablespaces, and the database's schema objects (such as tables, views, synonyms, indexes, sequences, clusters, and stored procedures).

TABLESPACES are used to group all of an application's objects to simplify certain administrative operations. A tablespace can be set *on-line* (accessible) or *off-line* (not accessible). The DIIDB database has many tablespaces. The original tablespace name SYSTEM is created during ORACLE installation. The SYSTEM tablespace cannot be set off-line or removed.

All data in a tablespace is stored in segments. A SEGMENT is set of extents allocated for a logical structure. An EXTENT is a specific number of contiguous data blocks, obtained in a single allocation, used to store a specific type of information. A DATA BLOCK corresponds to a specific number of bytes of physical database space on disk. The DATA BLOCK size is unchangeable after database creation.

Most segments begin at some specified size (number of extents), and grow dynamically (adding extents), as required. The DBA can monitor the space usage (extents) of a tablespace by looking at these views in SQL*Plus using these commands:

```
select * from sys.dba_segments;
select * from sys.dba_extents;
select * from sys.dba_free_space;
```

NOTE: To display the structure of each view above, enter "DESC table_name" (e.g., DESC sys.dba_extents).

SCHEMA OBJECTS are logical structures that refer directly to the database's data. Schema objects include tables, views, synonyms, sequences, indexes, cluster, program units, and database links.

A TABLE is the basic unit of data storage in an ORACLE database. Table data is stored in rows and columns. Each table is defined with a table name and a set of columns. Each column is defined with a column name, a data type (Character, Varchar2, Number, and Date), and a width. A VIEW is a database object that is treated like a table. However, views do not actually contain or store data. Whenever the view is queried, it derives data from the base tables on which the view was defined. Hence, a view can also be considered a *stored query*.

A SYNONYM is an alias for a table, view, sequence, or program unit. It is used to provide public access to an object, or to provide location transparency for tables, views, or program units on a remote database. A synonym can be either *public* (available to all users) or *private* (available only to a single user).

A SEQUENCE is used to generate a serial list of unique numbers for numeric columns of a database table.

INDEXES are optional structures associated with tables and clusters. Indexes are created to increase the performance of data retrieval and enable the search of records in sorted order or the random access of particular records based on a user-specified key. This minimizes the number of accesses by the program and can increase the efficiency of the database. An index can be created on one or more columns of a table. Indexes are logically and physically independent of the data in the associated table. They can be dropped or created at any time with no affect on the tables or other indexes. Indexes can be unique or non-unique. Unique indexes guarantee that no two rows in a table have duplicate values in the columns that define the index, while non-unique indexes do not impose this restriction on column values.

A CLUSTER is a group of tables that share the same data blocks because they share common columns and are often used together. They are used to improve disk access time and data retrieval.

A PROGRAM UNIT refers to stored procedures, functions, and packages. A stored PROCEDURE or FUNCTION is a set of Structured Query Language (SQL) and Procedural SQL (PL/SQL) statements grouped together as an executable unit to perform a specific task. A PACKAGE is used to encapsulate and store related procedures, functions, and other packages constructed together as an executable unit in the database. A DATABASE TRIGGER is a stored procedure that is implicitly executed (fired) when an INSERT, UPDATE, or DELETE statement is issued against the associated table.

A KEYWORD is a named object that describes a path from one database to another. It is implicitly used to access data on the remote database.

4. Database User Access and Privileges

Privileges provide control and management of access to a database. Before privileges can be granted to a user, a UNIX account and an ORACLE account must have been created (Section 9 details these procedures). Privileges can be assigned in two different ways:

DIRECTLY TO USERS

Privileges can be granted to a user explicitly (directly). For example, a privilege to delete a record from table X_TEST in the database can be granted explicitly (directly) to the user SMITH.

ROLES

A role is a collection of data access privileges, system privileges, and/or roles, which can be granted to and revoked from users. Privileges can be granted to *roles*, and each role can be granted to one or more users. For example, a privilege to insert a record in the COUNTRY_CODES can be granted to the role DIIDB_USER, which in turn can be granted to the users SMITH and JOHN. A privilege to execute a particular application can be granted one or more roles, and these roles can then be granted to the appropriate users. Roles facilitate the maintenance of privileges.

Roles can be provided by the DBA for individuals and groups of users for easy and controlled privilege management such as reducing privileges, dynamic privilege management, selective availability of privileges (enable or disable), application awareness, and application-specific security. A role can also be granted to other roles, which in turn can be granted to one or more users. Roles can be enabled or disabled at the user level.

The highest level of privileges and access to a database must be limited to the DBA(s). To access a database, ordinary users must have a valid ORACLE user name and password, must have the "connect" role (a collection of privileges), and may or may not have the resource and/or DBA roles that are granted by the DBA.

The CONNECT role allows a user to connect to a database and view information in the ORACLE data dictionary. A user who possesses only the connect role cannot create, alter, or drop tables, indexes, views, synonyms, clusters, or sequences.

The RESOURCE role allows a user to create, alter, and drop tables, indexes, views, synonyms, clusters, and sequences. With the resource role, a user can grant or revoke schema objects to and from other users. A user with the resource role must also have the connect role.

The DBA role allows users to:

- Access any user's data and perform any SQL statement upon it
- Grant and revoke database system privileges
- Create public synonyms, public database links, roles, and user accounts
- Control system-wide auditing and table-level auditing defaults
- Perform database exports and imports
- Perform database-wide maintenance operations such as adding tablespaces, data files, setting tablespace on- or off-line, backing up tablespaces, and archiving log files.

The IMP_FULL_DATABASE role allows users to select any table, back up any table, and insert, delete, and update certain system tables.

The EXP_FULL_DATABASE role allows users to log on as other users when performing full database imports.

5. ORACLE Database Startup and Shutdown

An ORACLE database is not always available to all users. To have control over the current status of an ORACLE database, only the DBA can start up or shut down the ORACLE database. When a database is open, users can access the information in it. When a database is closed, users cannot access the information. This is desirable, at times, to prevent users from corrupting the database while diagnostic and maintenance procedures are being carried out.

For specific instructions, refer to Database Startup and Database Shutdown.

6. Database Recovery and Backup

Database recovery and backup enables the restoration of damaged data and/or control files. If hardware, software, network, process, or system failure occurs, the database must be recovered as quickly as possible so normal operations can be resumed. Several problems can halt the normal operation of an ORACLE database or affect writing database information to the disk. The most common types of failure are:

USER ERROR FAILURE is caused by users, such as when someone accidentally drops a table.

STATEMENT FAILURE occurs when there is a logic failure in the handling of a statement in an application program. An error code or message is returned.

PROCESS FAILURE is the failure of the user, server, or background process in a database instance (e.g., an abnormal disconnect or process termination).

INSTANCE FAILURE occurs when a problem arises that prevents a database instance from continuing to work. This failure can result from either a hardware problem (such as a power outage) or a software problem (such as an operating system crash). An instance recovery is automatically performed as part of the next instance startup.

NETWORK FAILURE occurs when communication networks (such as the local area network, phone lines, etc.) are aborted (disconnected) on a distributed database system. This failure can interrupt the normal operations of a database system.

MEDIA FAILURE is a physical, non-recoverable error which can arise when trying to read or write a file that is required to operate a database. For example, a disk head crash causes the loss of all files on a disk drive.

6.1 Database Recovery

Several different features give the DBA flexibility when recovering databases:

ROLLING FORWARD reapplies all changes recorded in the redo log to the data files.

ROLLING BACK reverses the uncommitted database transactions recorded in the rollback segments and returns the database to a known stable point while the database is running.

ARCHIVING saves data found in the on-line redo logs for possible later use while the database is running. This feature protects the database from disk failure, providing for complete recovery right up to the moment before failure.

An IMAGE BACKUP performs block-by-block copying while the database is shut down. Image backups can be done on individual tablespaces as well. If required, this type of physical backup can recover the status of the entire database.

The EXPORT utility enables a logical backup of selective data or an entire database to a file while the database is running. The exported files can be saved on disk or archived to tape.

The IMPORT utility enables restoration of logical data to the database from an exported file while the database is running.

6.2 Database Backup

Database backups safeguard against potential media failures that can damage files. Routine, periodic backups provide the best protection against data loss and corruption's. Database backups can be performed whether a data file is on- or off-line**.**

6.3 Off-Line Database Backups

Any data file can be backed up when the database is shut down or a tablespace is off-line. Database backup and recovery are available via a menu-driven interface under the RECOVERY directory. The general procedures for off-line backups are:

a. Shut down the database using either shutdown immediate or shutdown abort procedures as described in the Database Shutdown Section.

b. Compress all ORACLE database files. Application database files are not stored under $ORACLE_HOME:

```
compress APPLICATION_1/*.dbf
compress APPLICATION_2/*.dbf
```

c. Copy all data files, on-line redo log files, and control files to tape by following these procedures:

1. tar the ORACLE database files to tape using the applicable command for your site. The following commands assume a DAT tape is being used:

(1) (For an HP)

```
tar cvf /dev/rmt/3m APPLICATION_1/*.dbf.Z
tar cvf /dev/rmt/3m APPLICATION_2/*.dbf.Z
```

OR

```
cpio -ocBuv APPLICATION_1/*.dbf.Z>/dev/rmt/3m
cpio -ocBuv APPLICATION_2/*.dbf.Z>/dev/rmt/3m
```

(2) (For Sun)

```
tar cvf /dev/rmt/0bmn $APPLICATION_1/*.dbf.Z
tar cvf /dev/rmt/0bmn $APPLICATION_2/*.dbf.Z
```

OR

```
cpio -ocBuv $APPLICATION_1/*.dbf.Z>/dev/rst0
cpio -ocBuv $APPLICATION_2/*.dbf.Z>/dev/rst0
```

2. tar the *$ORACLE_HOME/dbs* directory to tape using the applicable command. The following commands assume a DAT tape is being used:

   (For Sun)
   ```
   tar cvf /dev/rmt/0bmn $ORACLE_HOME/dbs
   ```

3. tar the */h/data/global* directory to tape using the applicable command. The following commands assume a

   DAT tape is being used:

   (a) (For an HP)
   ```
   tar cvf /dev/rmt/3m /h/data/global
   ```
   (b) (For Sun) (8mm tape)
   ```
   tar cvf /dev/rmt/0bmn /h/data/global
   ```

4. To preserve any messages that have been created, tar the */h/USMTF/data* directory to tape (do this for each machine):

   (a) (For an HP)
   ```
   tar cvf /dev/rmt/3m /h/USMTF/data
   ```
   (b) (For Sun)
   ```
   tar cvf /dev/rmn/0bmn /h/USMTF/data
   ```

NOTE: See the Sun Solaris or HP Reference Manual for more details about the UNIX ***tar*** and ***cpio*** commands.

---

5. Repeat Steps 1, 2, 3, and 4 to ensure that there is a backup for each tape.

d. Restart the database:

1. Uncompress the ORACLE database files:

   ```
   uncompress $APPLICATION_1/*.dbf.Z
   uncompress $APPLICATION_2/*.dbf.Z
   ```

2. Start up the ORACLE database.

3. Start up the AUDIT TRAIL, QUEUE MANAGER, and tti.

6.4 On-Line Database Backups

An on-line backup can be performed while the database is running. Thus, the DBA need not shut down the database to archive data. Data that is being accessed can also be archived during an on-line backup.

ORACLE can be operated in two archiving modes: *archived* and *noarchived*:

ARCHIVED MODE

When a database is operating in the *archived* mode, the on-line redo logs are saved before they are overwritten and the most recent backup can be used as part of data recovery. After restoring the necessary data files from the backup, database recovery can continue by applying archived and current on-line redo log files to bring the restored data files current. The files assembled by a full backup can be used to restore damaged files as part of database recovery from disk failure.

NOARCHIVED MODE

When a database is operating in the *noarchived* mode, the redo log files are overwritten without being archived and the most recent backup can be used to *restore* (not recover the database). Because the archived redo log files are not available to bring the database current, all database work performed since the database backup must be repeated. A full backup is the only method available to partially protect the database against disk failure.

The ON-LINE REDO LOG consists of at least two files that store all changes made to the database as they occur: one is optionally being spooled while the other is being written. These files are re-used once they fill up and are written to disk. At a minimum, archived redo logs that date back to the beginning of the oldest usable off-line database backup must be saved. The latest on-line redo log must also be saved. Archived redo logs previous to the last full backup can either be moved to tape or deleted.

7. Accessing ORACLE Server Manager

The ORACLE Server Manager is a DBA-management utility that performs these tasks:

- ORACLE instance starting and stopping
- ORACLE database mount, dismount, open, and close
- Monitoring of ORACLE database real-time use and performance
- Backup and recovery of database logs and data
- SQL statement execution
- PL/SQL statement execution.

Functions are selected from pull-down menus or initiated with commands from the operating system prompt. Server Manager can be invoked from either line mode (svrmgrl) or menu mode (svrmgrm). The menu mode is not discussed here. See Database Startup and Database Shutdown sections for specific examples of how to use ORACLE Server Manager. To invoke Server Manager in line mode:

a. Enter:

svrmgrl

<Return>. The SQL/DBA> prompt is then displayed.

b. For PL/SQL statement execution, enter:

connect <username> / <password>

<Return>, and the statement or series of statements to be executed..1.5 Accessing ORACLE DBA Utilities.1.5 Accessing ORACLE DBA Utilities.1.5 Accessing ORACLE DBA Utilities.1.5 Accessing ORACLE DBA Utilities.1.5 Accessing ORACLE DBA Utilities

For all other DBA functions, enter "connect internal" <Return>, and the command(s) necessary to perform the intended operation(s). Refer to the current revision of the ORACLE Server Manager User's Guide for further information.

NOTE: SQL commands can be entered in either upper- or lower-case.

NOTE: During a shutdown or startup of the database server, ORACLE shutdown and startup is handled by scripts invoked by the operating system.

8. Database Startup

Follow these steps to start up a database:

a. Log onto the database server if necessary.

b. Invoke SQL*DBA:

1. Enter:
   sqldba lmode=y

   or

   mode=line <Return>.

2. At the SQLDBA> prompt, enter:

   connect internal <Return>.

3. Enter:

startup <Return>.

(ORACLE will automatically start an instance, mount, and open the database.) Observe the display of messages acknowledging the startup of the database.

4. To quit SQLDBA when done: enter

   Exit.

9. Database Shutdown

An ORACLE database can be shut down using one of three options:

SHUTDOWN NORMAL
ORACLE waits for currently enrolled users to disconnect from the database, prohibits new users from logging, closes and dismounts the database, and shuts down the instance. Shutdown normal is accomplished via the "shutdown" or "shutdown normal" command.

SHUTDOWN IMMEDIATE
ORACLE immediately terminates any current client SQL statement being processed (does not wait for users currently connected to the database to disconnect) and rolls back the uncommitted statements. (This option should be used when a reboot or power shutdown is anticipated, or when the database is functioning irregularly.) Shutdown immediate is accomplished by entering "shutdown immediate" instead of "shutdown."

SHUTDOWN ABORT
Following shutdown abort option, the database instance is aborted immediately, uncommitted transactions are not rolled back, and the next startup of the database will require instance recovery procedures (automatically performed during database startup). It is not as violent as the name implies, and it ensures the database shuts down.. Shutdown abort is accomplished by entering "shutdown abort" instead of "shutdown."

These steps are required to shut down an open database:

a. Enter:

   sqldba lmode=y <Return>.

b. Enter

   connect internal <Return>.

c. Enter

   shutdown normal

   or

   shutdown immediate

   or

   shutdown abort <Return>.

   (ORACLE will automatically close and dismount the database and shut down the instance.)

   Observe the display of messages acknowledging the   startup of the database.

d. To quit sqldba when done, enter:

   Exit.

10. Accessing SQL*PLUS

SQL*Plus is an ORACLE tool that enables users to use Structured Query Language (SQL) or Procedural Language/Structured Query Language (PL/SQL) to query, update, delete, create, and modify database tables. SQL*Plus is usually invoked from a script using the following command line:

```
sqlplus{user name and password}
```

To invoke SQL*Plus interactively, enter:

sqlplus at the operating system prompt,

{username}

at the username prompt (where {username} is a valid ORACLE account name), and

{password}

at the password prompt (where {password} is the password associated with the user name provided).

The sql> prompt is displayed, enabling the entry of SQL commands. Refer to the current revision of the *ORACLE SQL Language Reference Manual* or *SQL*PLUS* `User's Guide` for more information on SQL commands.

NOTE: User access to SQL*PLUS is a data security problem. DII COE users should not have access to SQL*PLUS or any COTS database browser, because they have DII COE Core Database access, and the use of SQL*PLUS would allow them to bypass the security measures incorporated in the DII COE applications.

NOTE: SQL commands can be entered in either upper- or lower-case.

11. Passwords

Each ORACLE database user account requires both an ORACLE user name and password. User accounts are created by the DBA. As required, the DBA can change privileges for any user. For security and monitoring purposes, the ORACLE user account is tied to the UNIX account. Thus, a separate ORACLE password is not required and is not provided.

In DIIDB, user names are created as "identified externally." These are also referred to as OPS$ accounts. This allows for automatic logins to the database during which ORACLE reads the UNIX operating system user account to authenticate each user's status. This process eases user logon into ORACLE. ORACLE users can be externally logged onto ORACLE-based applications. Thus, an application or tool can be invoked without having to manually enter <username> and <password> each time the database is accessed. The following four different scripts are used with regard to passwords:

/h/cots/*ORAS/bin/alter_user_passwd.csh*, /h/cots/*ORAS/bin/create_external_user.csh*, */h/cots/ORAS/bin/create_internal_user.csh*, /h/cots/*ORAS/bin/drop_user.csh*.

Example: to invoke *sqlplus* on an Xterm from within a specific user account, enter: "sqlplus /", then the sql> prompt appears.

12. Maintaining SQL*Net

SQL*Net is ORACLE's remote data access software and enables client/server communications across the network. DIIDB is currently running both SQL*Net Version 1 and SQL*Net Version 2. Version 1 is available only for any backwards-compatibility situations and long-term support is not guaranteed.

For an application on a client machine to communicate with the ORACLE database, there must exist a listener to field the requests and forward them to the specified database SQL*Net V2 uses a listener process, tnslsnr, to support client requests for database services. The listeners run only on the database server. SQL*Net supports the TCP/IP Network Protocol.

In order to run the several SQL*Net tools described below:

su - oradba

12.1 SQL*Net V2.1

The default ORACLE network connection for DIIDB is SQL*Net V2.1. Sourcing the ORACLE environment, therefore results in (among other things)

TWO_TASK = dii.world

where dii.*world* is a connection string that identifies the local ORACLE database server. This is described in greater detail below. To determine whether SQL*Net V2 is running on the database server:

lsnrctl stat

Alternately:

ps -ef | grep tnslsnr

where *tnslsnr* is the name of the V2 listener on the database server machine. If it becomes necessary to start the V2 listener:

lsnrctl start

To stop the V2 server:

lsnrctl stop

There are three ascii files associated with the V2 software. On the DIIDB database server they are stored in the */var/opt/oracle* directory. These are:

listener.ora
tnsnames.ora
sqlnet.ora

For the client machines, only *tnsnames.ora* and *sqlnet.ora* exist under */var/opt/oracle*. If it becomes necessary to edit these files, a UNIX editor such as *vi* can be used. One must be careful not to inadvertently add or delete any "extra characters" in these files, in particular, trailing parentheses. These files can also be maintained via the GUI tool *Netman*, which is discussed below.

Examples of valid V2 connection strings are:

dii.world
nmcc.smil.mil

The default DII connection string, that value to which TWO_TASK is set, is dii.*world*, which references the local database server machine. The file */var/opt/oracle/tnsnames.ora* defines the connection information.

A connection string, also called a service name, maps to a connect description stored in the network configuration file *tnsnames.ora*. A connect description is a specially formatted description of the destination for a network connection, consisting of sets of keywords and values. It lists the communities of which the client is a member, the community protocol, e.g., TCP/IP, host name (or alternately, IP address) and the UNIX port number allocated. The *tnsnames.ora* file resides on each client and database server machine. If connection data changes them, each copy of this file should be updated. This onerous task will be eventually (or soon) replaced by a centralized maintenance scheme.

13. Full System Import

Import is the complementary utility to Export. Import reads files created by Export and writes the data back into a database. When importing a file, you have some flexibility about which data to import from the file.

Some circumstances may call for the re-creation of a database, due to the loss of files required for operating a database. One example is the loss of all control files, although in this case there are also other alternatives.

To perform a full database import, follow these steps:

1. Locate all export files containing data you wish to import. You may use one file containing all database data, or several export files of different types (full database, incremental, cumulative, or individual user). You can import directly from tape.

2. Close the database.

3. Shut down all instances accessing it.

4. Edit the INIT.ORA file if necessary.

5. Follow the following database creation steps:
    e. Back-up existing database
    f. Create an initialization (Init.ora) file
    g. Edit the new initialization file
    h. Edit all existing initialization files
    i. Start SQL*DBA
    j. Issue the SHUTDOWN Command
    k. Issue the STARTUP NOMOUNT Command
    l. Issue the CONNECT INTERNAL Command

m. Issue the CREATE DATABASE Statement
n. Create the Data Dictionary for the New Database
o. Copy the Initialization file to Client workstations
p. Back Up the New database files

Depending on the type of export you performed, you may have to run CATALOG.SQL. See the *ORACLE Server Utilities User's Guide* for more information on export types. When database creation is complete, the database is left open.

6. Perform an import using the desired export file(s).

14. Full System Export

To export the entire database, for whatever reason, including backup or to populate another machine, the following file can be created as described below:

File exp_full.par:

```
userid = system
full = Y
compress = N
file = exp_full.dmp
log = exp_full.log
```

To run, go to the directory where the export will reside and enter:

exp parfile=exp_full.par

You will be queried for the ORACLE "system" password. *oradba* will need privileges to write into that directory. Allow sufficient storage for the export file.

15. SQL*Loader Utility

SQL*Loader is a product for moving data from external files into tables in an Oracle database. It has many features that add power and flexibility. SQL*Loader loads data in a variety of formats, performs filtering (selectively loading records based upon the data values), and loads multiple tables simultaneously. During execution, SQL*Loader produces a detailed *log file* with statistics about the load. It may also produce a *bad file* (records rejected because of incorrect data) and a *discard file* (records that did not meet your selection criteria).

You must provide two types of input to SQL*Loader to load data from external files into an Oracle database: the data itself and control information describing how to perform the load. Data to be loaded into the Oracle database must exist in files on disk or on tape. These datafiles require interpretation to be loaded by SQL*Loader. This information is found in the control file.

You must provide a file called the control file as an input to the SQL*Loader. The control file tells SQL*Loader how to interpret the datafile.

You may load data in various formats. It is usually read from one or more datafiles. However, the data may also be placed in the control file after the control file information.

15.1 Conditions for Loading

To load data with a conventional load, the tables to receive the data must already exist in the database. There are no special requirements for these tables. The tables may be clustered or indexed, or they may actually be a view for which you have insert privileges. Tables may already contain data, or they may be empty.

The following privileges are required for a conventional load:

1. You must have INSERT privileges on the table to be loaded.
2. You must have DELETE privileges on the table to be loaded, when using the REPLACE option to empty out the table’s old data before loading the new data in its place.
3. You must have write access to all labels you are loading into a Trusted Oracle7 database.

16. Creating New User Tables

Each DII COE user has his/her own account in the ORACLE database. Users can create tables using SQL*Plus if they have access to it, usually via an xterm. The new tables are stored in the user's default tablespace. The *ORACLE SQL Language Reference Manual* and the *ORACLE SQL*Plus User's Guide* provide more information on the available commands.

17. Database Pulldown Monitor Routines from svrmgrm

The ORACLE routines described in this appendix are used to monitor the state of the database. They can help to avoid problems and aid in troubleshooting errors, especially when problems occur while adding users. The reports generated by these routines indicate when the system tablespace in full.   Adding another file to the system tablespace can often correct these problems.

These utilities are very useful when a tablespace becomes too full or when a database object attempts to exceed its MAX_EXTENTS. The database can be monitored daily and status reports can be produced. This process enables the DBA to spot and eliminate many potential problems before they occur. Also, the routines enable the DBA to regulate and schedule some of the database "repair work."

These c shell routines can be run daily by placing them in the DBA's *crontab* file. When this is done, the routines will access the database at the designated time and will create an output file containing useful information about database storage space. The script file can concatenate the information files and e-mail them to interested parties. The programs were designed to be run weekly, but they can easily be run daily. It requires one month to accumulate a usable amount of data in the CHANGED column of the reports.